Paper 302-2010

What's New in SAS/ACCESS®

Howard Plemmons, SAS Institute Inc., Cary, NC

ABSTRACT

The "What's New" paper for SAS/ACCESS highlights new product development and new development in existing SAS/ACCESS products and components. This paper highlights new engine development efforts, which include new SAS/ACCESS products—SAS/ACCESS® Interface to Aster nCluster, SAS/ACCESS® Interface to Greenplum, and SAS/ACCESS® Interface to Sybase IQ. Significant development efforts in existing SAS/ACCESS products and components are also highlighted, including SAS/ACCESS® Interface to DB2 on z/OS and the PC Files Server component of SAS/ACCESS® Interface to PC File Formats on Windows.

INTRODUCTION

The purpose of this paper is to provide details about new SAS/ACCESS engine development and new development in existing SAS/ACCESS products. The contents provide product overviews and functional details of new SAS/ACCESS products, features, and functionality. For a more complete understanding, see SAS/ACCESS documentation, which can be found on the SAS Technical Support Web site at http://support.sas.com. Select **Documentation** from the **Knowledge Base** section on this Web site.

SAS/ACCESS is one of the fundamental components of the SAS system for accessing DBMS data. Basically, the functionality of SAS/ACCESS enables SAS to communicate with a DBMS. The communication strategies used by SAS/ACCESS range from totally transparent within SAS to directly controlled by the user. An example of total transparency is shown in the example code below:

```
/*--- connect to the DBMS ---*/
libname x <dbms> <connection options>;
data a;
  set x.dbms_tab;
run;
```

In this example, a SAS user is unaware of the SQL and database interaction initiated on his behalf. The SAS code connects to the DBMS, reads all of the data from the DBMS table named dbms_tab, and stores it in a SAS WORK data set named work.a.

Direct control by the user is shown in the example code below:

```
proc sql;
connect to <dbms> ( <connection options> );
  select * from connection to <dbms>
  ( <DBMS SQL Select statement> );
  exec ( <DBMS DDL statement> ) by <dbms>;
quit;
```

In this example, a SAS user has control over the SQL pushed to the DBMS. He has control over specific DBMS interactions and DDL statements, which enable him to interact directly with the DBMS. For example, the user could drop tables, create tables, manage indexes, and update and insert rows using the EXEC option in PROC SQL.

The syntax used in the previous examples might be familiar to you from previous SAS/ACCESS engine experience. The same syntax is used with the new SAS/ACCESS products. Using a standard SAS I/O framework enables a level of functional portability across many different DBMSs. A number of options are portable across all SAS/ACCESS products. To communicate these options and DBMS-specific features to the DBMS requires SAS options. SAS options enable the user to tell the DBMS about specific functionality that he wants to enable with SAS through a SAS/ACCESS engine.

The following diagram shows the control flow to the DBMS using SAS and SAS/ACCESS, where SAS is on one machine and the DBMS server is on another:

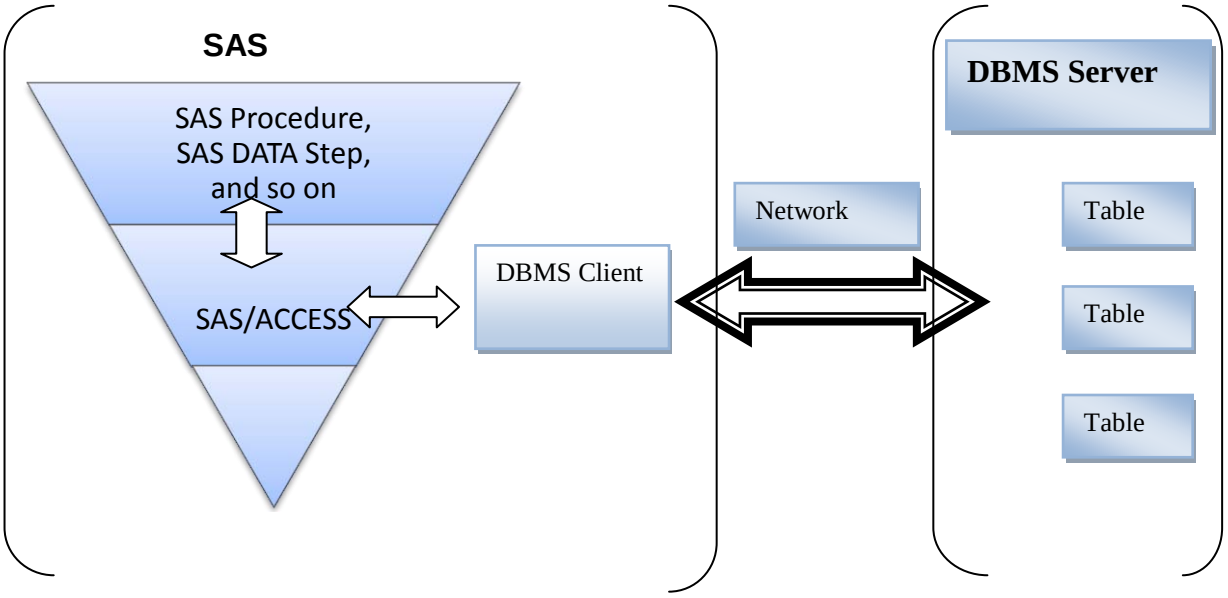

Figure 1. SAS Interaction with the DBMS

NEW SAS/ACCESS PRODUCT COMMON COMPONENTS

This section avoids repeating information about the new SAS/ACCESS products. Common components are an information focus, not a common set of features shared across databases. This section gives you a basis of understanding that might help you when reading the sections specific to SAS/ACCESS.

SUPPORT MATRIX BY HOST

Where do the new SAS/ACCESS products run? SAS is ported to and executes on many platforms and hosts. New SAS/ACCESS products are ported to hosts supported by SAS that have DBMS client and driver support. The matrix identifies SAS hosts that currently support the new SAS/ACCESS products.

Platform Matrix	HP PA-RISC 64-bit	Solaris 10 SPARC 64-bit	RS/6000 64-bit	HP-UX 11 64-bit	Windows XP x64 64-bit	Windows 32-bit	Linux AMD or Intel 32-bit	Linux AMD or Intel 64-bit	Solaris 10 AMD or Intel 64-bit
SAS/ACCESS Interface to Aster nCluster					X	X	X	X	
SAS/ACCESS Interface to Greenplum		X	X	X	X	X	X	X	X
SAS/ACCESS Interface to Sybase IQ	X	X	X	X	X	X	X	X	X

Figure 2. New Product Availability

Versions of the platform and supporting libraries might be specific to a DBMS vendor release. Review the DBMS client requirements and the SAS *System Requirements* before proceeding with the SAS DBMS installation and configuration process.

INSTALLATION OVERVIEW

The new SAS/ACCESS products—SAS/ACCESS Interface to Aster nCluster, SAS/ACCESS Interface to Greenplum, and SAS/ACCESS Interface to Sybase IQ—have a multiple step installation process. Some of the following information and hints might save you time when configuring SAS and the DBMS client.

First, what are the components that comprise the SAS/ACCESS product and where are they located?

Product	Location of SAS/ACCESS Engine	Location of Client Component	Location of Loader Component
SAS/ACCESS Interface to Aster nCluster	SAS Media	http://www.asterdata.com	http://www.asterdata.com
SAS/ACCESS Interface to Greenplum	SAS Media	SAS Media	SAS Media
SAS/ACCESS Interface to Sybase IQ	SAS Media	http://www.sybase.com	Included in Sybase IQ Client Component

Figure 3. Installation Overview

Second, what are some of the configuration issues and hints that apply to these new SAS/ACCESS products?

- The new SAS/ACCESS products are based on an ODBC interface to the DBMS. The SAS/ACCESS product has been specifically tuned for each DBMS and supports the DBMS vendor load and extract process from SAS.
- On UNIX, for SAS/ACCESS Interface to Aster nCluster and SAS/ACCESS Interface to Sybase IQ, there is a direct link to the vendor driver. A driver manager is provided with SAS/ACCESS Interface to Greenplum. Therefore, you do not need to install or use a separate driver manager with these new SAS products.
- To enable the SAS/ACCESS products, you must install and configure an ODBC client on the platform on which you intend to use SAS to access the DBMS. The SAS/ACCESS platform must appear in the supported matrix for your DBMS.
- You must install and configure the ODBC environment on Windows, UNIX, and Linux. You need to obtain DBMS connection information (for example, user name, password, port, server, and schema) before proceeding with the installation process. The SAS/ACCESS Interface to Greenplum ODBC driver is provided on the SAS media and is part of the SAS installation process.
- When you set up the ODBC driver on Windows, be sure to test the connection to the DBMS before using SAS. (The **Test** button is in the ODBC configuration window.)
- When you install the SAS/ACCESS products, be sure to run PROC NICKNAME. This procedure updates the SAS core catalog so you can use an engine alias name instead of the SAS image name. After installation has completed, you might want to consider the following example code as a validation test:

```
/*--- Sample test job for new SAS/ACCESS engines, should validate engine ---*/
/*--- and client functionality.                                          ---*/
libname x <your DBMS nickname> user=<your user id> password=<your pw> …;

data work.a;
   x=1;
   y='aaa';
   output;
run;

data x.newdbtab;
   set a;
run;

/*--- see some debug code with this option ---*/
options sastrace=',,,d' sastraceloc=saslog;

proc append base=x.newdbtab data=a (bulkload=yes);

/*--- turn off tracing ---*/
options sastrace=',,,,';
```

```
proc sql;
   select * from x.newdbtab;
quit;

proc sql;
   connect to <dbms nickname> ( <connection info> );
   select * from connection to <dbms nickname> ( select * from newdbtab );
quit;

proc delete data=x.newdbtab;
run;
```

For complete installation information, see SAS documentation.

COMMON COMPONENTS

SAS/ACCESS products provide a feature set that helps SAS interface with a DBMS. This feature set is common across DBMS products. However, feature implementation and execution are DBMS specific. The following features identify key components in the new SAS/ACCESS products.

LIBNAME Engine

is a major component of the SAS/ACCESS product. It involves an interface between SAS/ACCESS and SAS using a common I/O model. The SAS I/O model drives the SAS/ACCESS product based on user requests from SAS. The user is insulated from direct interaction with the DBMS. However, the user can affect SAS/ACCESS behavior with SAS options. Categories of options can be used in the SAS LIBNAME statement as SAS data set options and options specified using SAS procedures.

Implicit Pass-Through

identifies an interface between SAS and the DBMS. Using implicit pass-through enables the user to communicate with the DBMS using SAS SQL. SAS SQL entered by the user is parsed and textualized into DBMS SQL, based on DBMS-specific rules. This process enables the user to code SAS SQL that can be executed on many DBMSs. The following example code shows implicit pass-through:

```
libname x <dbms> <connection and process options>;
proc sql;
   select * from x.dbtab where col1 > 10;
quit;
```

Explicit Pass-Through

identifies an interface between SAS and the DBMS. Using explicit pass-through enables the user to communicate directly with the DBMS by using DBMS SQL and DDL statements. The following example code shows explicit pass-through:

```
proc sql;
   connect to <dbms> ( <connection info> );
   select * from connection to <dbms nickname>
       (select * from dbtab where col1 > 10);
   exec (create table dbtab2 as select * from dbtab where col1 > 25) by <dbms>;
quit;
```

The code in the parentheses is DBMS-specific SQL. It is sent as is to the DBMS for processing. Any errors in the SQL code are surfaced for the SAS user to correct.

Debugging

can be performed by viewing the interaction between SAS and a DBMS. The following options enable you to view the SQL that is generated and passed to the DBMS:

```
options sastrace=',,,d' sastraceloc=<SASLOG | <file> >;

options sql_ip_trace = ALL;
```

The first set of options are used by the SAS/ACCESS product. The second set controls debugging information from PROC SQL. SAS documentation provides a complete option list and descriptions.

SQL Dictionary

is an internal structure in the SAS/ACCESS product that maps SAS functions to DBMS functions. This functionality enables the transformation of SAS functions into equivalent DBMS functions. Using this functionality in implicit pass-through provides the SAS textualizer with mapping information. This information is used to translate SAS functions into DBMS functions during the formation of the DBMS SQL statement. The following code shows how to obtain the SQL dictionary map from the SAS/ACCESS product. It shows the contents of the SAS/ACCESS Interface to ORACLE SQL dictionary as it appears in the SAS log.

```
libname x <your dbms> <connection options>
SQL_FUNCTIONS_COPY=<libref.member|SASLOG>;

SAS Function Mappings provided by SAS ACCESS engine:
    SAS             DBMS
FUNCTION NAME   FUNCTION NAME
-------------   -------------
   ABS            ABS
   ARCOS          ACOS
   ARSIN          ASIN
   ATAN           ATAN
   CEIL           CEIL
   COS            COS
   COSH           COSH
   DATETIME       SYSDATE
   EXP            EXP
   FLOOR          FLOOR
   LOG            LN
   LOG10          LOG
   LOG2           LOG
   LOWCASE        LOWER
   SIGN           SIGN
   SIN            SIN
   SINH           SINH
   SOUNDEX        SOUNDEX
   SQRT           SQRT
   TAN            TAN
   TANH           TANH
   TRIMN          RTRIM
   TRANSLATE      TRANSLATE
   UPCASE         UPPER
   STRIP          TRIM
   BAND           BITAND
   ATAN2          ATAN2
   STD            STDDEV_SAMP
   VAR            VAR_SAMP
   COALESCE       COALESCE
   DAY            EXTRACT
   MONTH          EXTRACT
   YEAR           EXTRACT
   HOUR           EXTRACT
   MINUTE         EXTRACT
   SECOND         EXTRACT
   DTEXTDAY       EXTRACT
   DTEXTMONTH     EXTRACT
   DTEXTYEAR      EXTRACT
```

The list shows you how SAS is mapping SAS functions to DBMS functions. You can modify the list dynamically to add functions or to modify the mapping process. The following code shows how to include an additional dictionary table, which is appended to the internal table for the duration of the LIBNAME statement:

```
libname x <your dbms> <connection options> EXTERNAL_APPEND=<libref.member>;
```

In this example, the additional SAS to DBMS mapped functions are dynamically added to the end of the internal SAS/ACCESS product SQL dictionary. These functions are evaluated as part of the preparation step in converting SAS SQL to DBMS SQL. Using this mapping might enable more SQL to be pushed to the DBMS for processing.

SAS/ACCESS INTERFACE TO ASTER NCLUSTER

Aster Data is described as a massively parallel data-application server that provides access to massive data for analysis. SAS/ACCESS Interface to Aster nCluster uses the Aster ODBC driver and the Aster nCluster loader for data loading. For more information about Aster Data, see http://www.asterdata.com/product/index.php.

SAS/ACCESS INTERFACE TO ASTER NCLUSTER HIGHLIGHTS

The ability to effectively communicate with the Aster DBMS is predicated on understanding and using SAS/ACCESS functions and attributes. The controls that enable you to communicate with the Aster DBMS come from the SAS LIBNAME statement, the SAS data set, and SAS procedure options. Here are a few of the options that you can use to interact with the Aster DBMS. For a complete list and descriptions of Aster options, see SAS/ACCESS documentation.

ASTER LIBNAME AND DATA SET HIGHLIGHTS

LIBNAME

DIMENSION (default NO)

> This option specifies what type of Aster table you will create with SAS. If you do not specify DIMENSION=YES, then SAS creates a fact table. Otherwise, SAS creates a dimension table. If you create a fact table, you are required to specify a partitioning key.

INSERTBUFF and READBUFF (default 1)

> These options buffer data moving into and out of Aster. Potential performance improvements can be gained with proper tuning and setting.

SCHEMA (default user ID)

> This option specifies what schema you want to use when loading and accessing Aster DBMS data. The default schema is the user ID that you have used to connect to the database.

LOGIN_TIMEOUT (default 0)

> This option specifies how long you want to wait until timing out your connection to Aster.

QUERY_TIMEOUT (default 0)

> This option specifies how long you want to wait until your query is timed out by Aster.

Data Set

BULKLOAD (default NO)

> If this option is set to YES, and the Aster nCluster loader is installed and supported, then SAS will invoke the Aster nCluster loader and use it to add records to new or existing Aster tables.

BL_

> This option identifies a set of SAS data set options that enable you to pass bulk-load commands to the Aster nCluster loader. These options enable you to customize the bulk-load operation in Aster.

OBS (default ALL)

> This is a SAS option that is used by SAS/ACCESS Interface to Aster nCluster. It identifies the number of rows to extract from the Aster DBMS table. It is useful with huge Aster tables and controls the number of rows returned from an SQL execution.

PRESERVE_COL_NAMES (default NO)

> This option provides a quoted context for DBMS column names that contain embedded blanks, special characters, or reserved names. A YES value ensures that column names are valid by enclosing them in quotation marks in the SQL-generation step. This option can be specified in the LIBNAME statement as well.

Procedure

With SAS procedures, you can influence Aster processing using SAS LIBNAME and data set options that are processed by SAS/ACCESS Interface to Aster nCluster, such as limiting and aggregating rows. For example, if you want to view a segment of an Aster table, use the following code:

```
/*--- only extract department 12 for additional processing ---*/
```

```
libname x aster <connection options>;
data sasuser.dept12;
   set x.company(where = (dept_no=12));
run;

/*--- limit processing to the west coast ---*/
proc sql;
   create view sasuser.west_coast as
      select * from x.company where region='WC';
quit;
```

ASTER DATA LOADING HIGHLIGHTS

Loading data into Aster can be accomplished using different SAS methods. For example:

```
/*--- insert rows into Aster ---*/
libname x aster <connection options>;

/*--- create and load a new table ---*/
data x.new_tab;
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab;
run;

proc sql;
   insert into x.new_tab as select * from work.a;
quit;
```

Each of these methods loads data into the database. If you have a significant amount of data to add, or a data insert performance requirement, then you should consider the Aster nCluster loader. This loader can significantly speed up data loading. Using the loader requires you to set SAS data set options that trigger the bulk-load process in SAS/ACCESS Interface to Aster nCluster. For example:

```
/*--- bulkload rows into Aster ---*/
libname x aster <connection options>;

/*--- create and bulkload a new table ---*/
data x.new_tab (bulkload=yes bl_dbname=abc bl_host=aster1);
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab(bulkload=yes bl_dbname=abc bl_host=aster1);
run;
```

SAS/ACCESS INTERFACE TO GREENPLUM

Greenplum is described as the world's fastest and most scalable database based on software and commodity hardware systems. SAS/ACCESS Interface to Greenplum uses SAS for ODBC and loading, which is facilitated by the Greenplum bulk-data-loading process. For more information about Greenplum, see http://ww.greenplum.com.

SAS/ACCESS INTERFACE TO GREENPLUM HIGHLIGHTS

The ability to effectively communicate with the Greenplum DBMS is predicated on understanding and using SAS/ACCESS functions and attributes. The controls that enable you to communicate with the Greenplum DBMS come from the SAS LIBNAME statement, the SAS data set, and SAS procedure options. Here are a few of the options that you can use to interact with the Greenplum DBMS. For a complete list and descriptions of Greenplum options, see SAS/ACCESS documentation.

GREENPLUM LIBNAME AND DATA SET HIGHLIGHTS

LIBNAME

CONNECTION (default SHAREDREAD)

The default value indicates that all Read operations that access the DBMS tables in a single libref share a single connection. There are other options that help you control connection processes with the Greenplum DBMS.

DBCOMMIT (default 1000 INSERT, 0 UPDATE)

This option controls when commits are pushed to the DBMS. For example, with the default, commits occur after every 1000 rows inserted or after every UPDATE statement. If an insert fails, then the data is rolled back to the last commit. If you insert less than 1000 rows, then a commit occurs at the end of the insert process.

INSERTBUFF and READBUFF (default 1)

These options buffer data moving into and out of Greenplum. Potential performance improvements can be gained with proper tuning and setting.

SPOOL (default none)

This option identifies that the data needs to be spooled so that a second pass through the data is presented in the same order as the first. This is critical to multi-pass SAS procedures that require data presentation order.

UPDATE_MULT_ROWS (default NO)

This option indicates to SAS that updating multiple rows with an update operation is acceptable.

Data Set

BULKLOAD (default NO)

If this option is set to YES, the rows added to the Greenplum database will use the Greenplum bulk loader. The bulk-load process can be applied to new or existing tables.

BL_ (default none)

This option identifies a set of SAS data set options that enable you to pass bulk-load commands to Greenplum. These options enable you to customize the bulk-load operation in Greenplum. For example, you could use BL_REJECT_LIMIT to control how many load failures occur before processing is terminated.

DBFORCE (default NO)

If this option is set to YES, then rows that exceed the length of the DBMS column are inserted. Otherwise, NO means that rows that exceed the length of the DBMS column are not inserted. This enables you to control character data truncation when processing rows in the database.

QUERY_TIMEOUT (default 0)

This option indicates the number of seconds to wait before timing out the query. The default of 0 is interpreted as no time limit. This option can be specified in the LIBNAME statement as well.

Procedure

With SAS procedures, you can influence Greenplum processing using SAS LIBNAME and data set options that are processed by SAS/ACCESS Interface to Greenplum, such as limiting and aggregating rows. For example, if you want to view a segment of a Greenplum table, use the following code:

```
/*--- only extract department 12 for additional processing ---*/
libname x greenplum <connection options>;
data sasuser.dept12;
   set x.company(where=(dept_no=12));
run;

/*--- limit processing to the west coast ---*/
proc sql;
   create view sasuser.west_coast as
      select * from x.company where region='WC';
quit;
```

GREENPLUM DATA LOADING HIGHLIGHTS

Loading data into Greenplum can be accomplished using different SAS methods. For example:

```
/*--- insert rows into Greenplum ---*/
```

```
libname x greenplum <connection options>;

/*--- create and load a new table ---*/
data x.new_tab;
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab;
run;

proc sql;
   insert into x.new_tab as select * from work.a;
quit;
```

Each of these methods loads data into the database. If you have a significant amount of data to add, or a data insert performance requirement, then you should consider the Greenplum loader. This loader can significantly speed up data loading. Using the loader requires you to set SAS data set options that trigger the bulk-load process in SAS/ACCESS Interface to Greenplum. For example:

```
/*--- bulkload rows into Greenplum ---*/
libname x greenplum <connection options>;

/*--- create and bulkload a new table ---*/
data x.new_tab(bulkload=yes bl_options='<db specific options>');
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab(bulkload=yes …);
run;
```

SAS/ACCESS INTERFACE TO SYBASE IQ

Sybase IQ is described as a highly optimized analytic business intelligence engine. SAS/ACCESS Interface to Sybase IQ uses the Sybase IQ ODBC driver and implements bulk-loading through the Sybase IQ LOAD TABLE process. For more information about Sybase IQ, see http://www.sybase.com.

SAS/ACCESS INTERFACE TO SYBASE IQ HIGHLIGHTS

The ability to effectively communicate with the Sybase IQ DBMS is predicated on understanding and using SAS/ACCESS functions and attributes. The controls that enable you to communicate with the Sybase IQ DBMS come from the SAS LIBNAME statement, the SAS data set, and SAS procedure options. Here are a few of the options that you can use to interact with the Sybase IQ DBMS. For a complete list and descriptions of Sybase IQ options, see SAS/ACCESS documentation.

SYBASE IQ LIBNAME AND DATA SET HIGHLIGHTS

LIBNAME

DBCREATE_TABLE_OPTS (default none)

> This option specifies DBMS SQL syntax to be appended to the SYBASE IQ CREATE TABLE statement generated by SAS.

DIRECT_EXE (default none)

> When this option is set to DELETE, SAS is directed to pass the DELETE statement to the DBMS, which can significantly increase performance by executing the DELETE process inside the DBMS.

READ_ISOLATION_LEVEL (default RC (READ COMMITTED))

> This option specifies a DBMS-specific value that defines the degree of isolation of the current application process from other concurrently running applications.

REREAD_EXPOSURE (default NO)

If this option is set to YES, then the SAS/ACCESS product behaves like a random access engine for the scope of the LIBNAME statement. Be careful when setting the value to YES because a relational DBMS might return rows of different values. In a multi-user environment, other users might delete and insert rows that could have an adverse effect on your processes when using this option.

TRACE and TRACEFILE (default NO and none)

If this option is set to YES and a file is identified, then additional debugging information is written to the trace file. This is a feature only in Windows and it might be helpful when gathering information for technical support calls.

Data Set

BULKLOAD (default NO)

If this option is set to YES, the records added to new or existing Sybase IQ tables will use the Sybase IQ LOAD TABLE process.

BL_

This option identifies a set of SAS data set options that enable you to pass bulk-load commands to Sybase IQ. These options enable you to customize the bulk-load operation in Sybase IQ.

INGORE_READ_ONLY_COLUMNS (default NO)

This option specifies whether to ignore or include read-only columns when generating an SQL statement in SAS for inserts or updates. If you try to modify a read-only column, the modification will fail when pushed to the DBMS.

SASDATEFMT (default NO)

This option maps a DBMS column to a SAS format. For example, the syntax for this statement is SASDATEFMT= (<*dbms col*> = '<*sas date fmt*>'…). This option enables you to map a SAS date-and-time field to a form that is recognized by the DBMS.

Procedure

With SAS procedures, you can influence Sybase IQ processing using SAS LIBNAME and data set options that are processed by SAS/ACCESS Interface to Sybase IQ, such as limiting and aggregating rows. For example, if you want to view a segment of a Sybase IQ table, use the following code:

```
/*--- only extract department 12 for additional processing ---*/
libname x sybaseiq <connection options>;
data sasuser.dept12;
   set x.company (where=(dept_no=12));
run;

/*--- limit processing to the west coast ---*/
proc sql;
   create view sasuser.west_coast as
      select * from x.company where region='WC';
quit;
```

SYBASE IQ DATA LOADING HIGHLIGHTS

Loading data into Sybase IQ can be accomplished using different SAS methods. For example:

```
/*--- insert rows into Sybase IQ ---*/
libname x sybaseiq <connection options>;

/*--- create and load a new table ---*/
data x.new_tab;
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab;
run;

proc sql;
   insert into x.new_tab as select * from work.a;
```

```
quit;
```

Each of these methods loads data into the database. If you have a significant amount of data to add, or a data insert performance requirement, then you should consider the Sybase IQ bulk-loading method. This method can significantly speed up data loading. Using this method requires you to set SAS data set options that trigger the bulk-load process in SAS/ACCESS Interface to Sybase IQ. For example:

```
/*--- bulkload rows into Sybase IQ ---*/
libname x sybaseiq <connection options>;

/*--- create and bulkload a new table ---*/
data x.new_tab (bulkload=yes bl_options='<db specific options>');
   set work.a;
run;

/*--- append to an existing table ---*/
proc append data=a base=x.new_tab(bulkload=yes …);
run;
```

ONGOING PRODUCT SUPPORT

SAS continues to research and upgrade its existing product suite to meet ever-growing user needs and demands. Some key product initiatives for SAS 9.2 included upgrades to SAS/ACCESS Interface to DB2 on z/OS and a continued upgrade of SAS/ACCESS Interface to PC File Formats on Windows, specifically the PC Files Server. The following sections identify the key features and performance impact of these upgrades.

SAS/ACCESS INTERFACE TO DB2—NEW FEATURES AND PERFORMANCE IMPROVEMENTS

The features and performance improvements require no specific user interaction. However, SAS and SAS/ACCESS Interface to DB2 must be upgraded to the third maintenance release for 9.2 on z/OS. Once you have upgraded, you will see the following features and improvements when executing SAS/ACCESS Interface to DB2.

Enhanced Error Messages

DB2 error messages now contain more details. This level of detail should provide additional insight into the interaction between SAS and DB2. The following example shows an error message generated from an invalid request before (regular font) and after (bold font) enhancing the message:

```
libname x db2;

proc sql;
     drop table x.mytab;
quit;


SAS 9.2 log:

4          proc sql;
5               drop table x.mytable;
WARNING: File X.MYTABLE.DATA does not exist.
WARNING: Table X.MYTABLE has not been dropped.
6          quit;


SAS 9.2M3 log:

3          proc sql;
4               drop table x.mytable;

DB2 ERROR:
RESULT OF SQL STATEMENT:
DSN00204E MYID.MYTABLE IS AN UNDEFINED NAME
A DUMP OF THE SQLCA FOR THE GET DIAGNOSTICS REQUEST FOLLOWS:
          ROW NUMBER: 0 ERROR CONDITION: 1 REASON CODE: 0
          SQLCODE: -204 SQLSTATE: 42704 SQLERRP: DSNXOTL
```

```
WARNING: File X.MYTABLE.DATA does not exist.
WARNING: Table X.MYTABLE has not been dropped.
5          quit;
```

Enhanced Performance of Data Extraction

A buffering scenario produces a significant performance improvement when extracting data from DB2. The extent of data buffering is communicated to DB2 using the SAS/ACCESS READBUFF option (default 1 row). The SAS code and output show a 65% performance improvement between SAS 9.2 and the third maintenance release for 9.2.

```
/*--- SAS 9.2 execution ---*/
options sastrace=',,,sd';

proc sql;
     connect to db2 (readbuff=1);
     create table work.test as
     select * from connection to db2
     (select * from myid.mytab);
quit;

From the SAS log:

Summary Statistics for DB2 are:
Total row fetch seconds were:                      23.888685
Total SQL prepare seconds were:                     0.018044
Total seconds used by the DB2 ACCESS engine were    45.749788


/*--- SAS 9.2M3 execution ---*/
options sastrace=',,,sd';

proc sql;
     connect to db2 (readbuff=10000);
     create table work.test as
     select * from connection to db2
     (select * from myid.mytab);
quit;

From the SAS log:

Summary Statistics for DB2 are:
Total row fetch seconds were:                       8.278659
Total SQL prepare seconds were:                     0.000330
Total seconds used by the DB2 ACCESS engine were    28.347795
```

Extended SQL Function Support

The internal SQL dictionary in the third maintenance release for 9.2 is significantly improved. In the following SAS code, the functions listed are mappings that aid in the textualization of SAS SQL into DB2 SQL:

```
libname x db2 [ssid=<ssid>] SQL_FUNCTIONS_COPY=SASLOG;
```

Database Read-Only Access Support

A new run-time option—DB2UPD (default YES)—has been added to enable read-only access to DB2 using SAS/ACCESS Interface to DB2. If you set DB2UPD=NO, then you are not able to update DB2 tables using SAS. Setting this option does not modify your DBMS security settings or privileges. However, it does provide an update block in SAS/ACCESS Interface to DB2.

DB2 Explain Support

You can now access the DB2 Explain facility using parameters in the SASTRACE option. The parameter E in SASTRACE tells SAS to output an explanation. The parameter X tells SAS to output an explanation and execute the SQL statement. The DB2 PLAN_TABLE must exist for the user who is running SASTRACE so that output can be collected and presented in the table.

The following code shows how to invoke and collect information:

```
DB2 NOTE:  The QUERYNO for the EXPLAIN is 2033622


The following code shows how Explain data can be collected through SAS:
libname x db2 schema=dsn8910;

options sastrace=',,,e';

proc print data=x.emp;
     where empno = '000010';
run;


1          libname x db2 schema=dsn8910;
NOTE: Libref X was successfully assigned as follows:
      Engine:        DB2
      Physical Name: DB2
2
3          options sastrace=',,,e';
4
5          proc print data=x.emp;
6               where empno = '000010';
7          run;

DB2_1: Explained on connection 0
EXPLAIN ALL SET QUERYNO=2033662 FOR SELECT "EMPNO", "FIRSTNME", "MIDINIT", "LAST
NAME", "WORKDEPT", "PHONENO", "HIREDATE", "JOB", "EDLEVEL", "SEX", "BIRTHDATE",
"SALARY", "BONUS", "COMM" FROM DSN8910.EMP WHERE ("EMPNO" ='000010' ) FOR FETCH
ONLY

WARNING: (ACCDB2M052W) The statement will not be executed because the current
setting of the EXPLAIN prevents that.
```

Here is an example of how the explanation can be extracted:

```
libname user db2;

%let queryno = <QUERYNO>;

data work.basic;

     set user.plan_table

         (keep=queryno qblockno parent_qblockno method creator tname

               table_type accesstype qblock_type);

     where queryno=&queryno;

run;

data work.extended;

     set user.plan_table

         (keep=queryno qblockno accesstype matchcols accesscreator

               accessname indexonly prefetch access_degree

               join_degree parallelism_mode join_type);

     where queryno=&queryno;

run;

data work.sort;

     set user.plan_table
```

```
        (keep=queryno qblockno method table_type sortn_uniq
              sortn_join sortn_orderby sortn_groupby sortc_uniq
              sortc_join sortc_orderby sortc_groupby);
    where queryno=&queryno;
run;
```

SAS/ACCESS INTERFACE TO PC FILE FORMATS—NEW FEATURES AND PERFORMANCE IMPROVEMENTS

The PC Files Server is a component of SAS/ACCESS Interface to PC File Formats. For more information about this component, see SAS/ACCESS Interface to PC File Formats documentation. The PC Files Server is a lightweight server that runs in a Windows environment and facilitates data access from PC files on UNIX or in other Windows environments. This server moves data from a specific Windows environment to a UNIX or Windows platform for analysis. The server supports additions and updates to server-based data. The following diagram provides an overview of the UNIX data flow process:

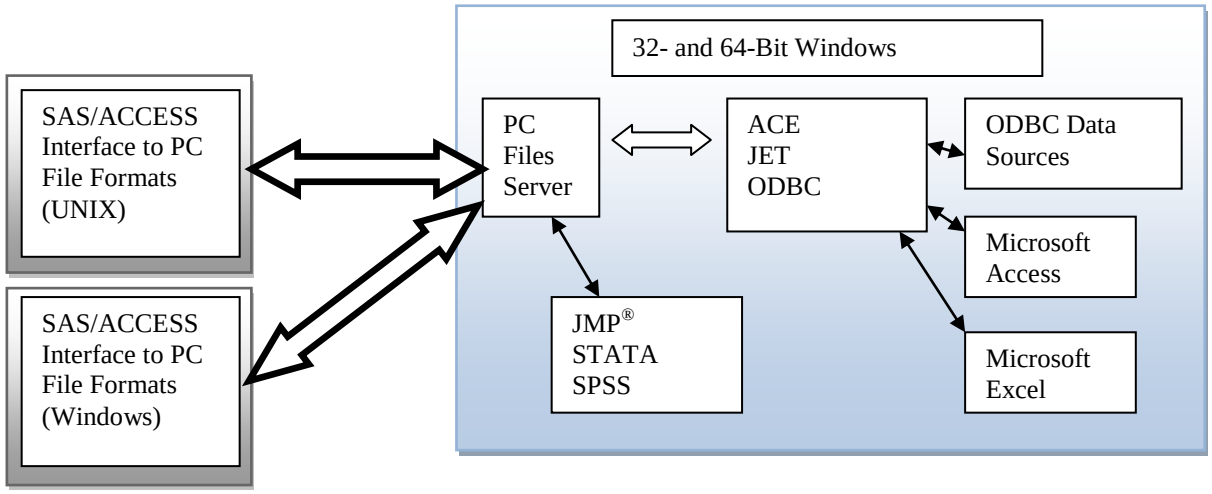

Figure 5. Data Flow through the PC Files Server

The data flow shows several types of data for the PC Files Server to export. Besides JMP, the IMPORT and EXPORT procedures process STATA and SPSS data. Beginning with the third maintenance release for 9.2, JMP access is available when using the IMPORT and EXPORT procedures in Base SAS software. The following SAS code running on UNIX or Windows interacts with the PC Files Server for data access:

```
/*--- get JMP data from the PC Files Server ---*/
proc import
    dbms=pcfs
    datafile="c:\db.jmp" out=a replace;
    server=fs;
run;

/*--- get EXCEL data from the PC Files Server ---*/
proc import
          dbms=excelcs
          datafile="c:\db.xls" out=b replace;
   server=fs;
run;

/*--- write JMP data to windows using the PC Files server ---*/
proc export
    dbms=pcfs
          outfile="c:\newjmp.jmp" data=a replace;
```

```
    server=fs;
run;
```

The ability to move data from Windows and other platforms has been the purpose of the PC Files Server for some time. However, performance issues hampered the effectiveness of the server. A goal for the third maintenance release for 9.2 was to improve extract and load performance when using the PC Files Server. The following tables show that the goal was met and significantly exceeded.

Microsoft Excel data imports and exports based on improvements in the third maintenance release for 9.2:

Microsoft Excel	**20 Columns**	**50 Columns**	**200 Columns**
Import 1,000 Rows	2X	3X	4X
Import 10,000 Rows	4X	4X	4X
Import 65,000 Rows	3X	3X	3X
Export 1,000 Rows	7X	7X	4X
Export 10,000 Rows	11X	9X	4X
Export 65,000 Rows	6X	5X	4X

Microsoft Access data imports and exports based on improvements in the third maintenance release for 9.2:**Microsoft Access**	**20 Columns**	**50 Columns**	**200 Columns**
Import 1,000 Rows	2X	4X	ACE Driver Limitation
Import 10,000 Rows	9X	7X	ACE Driver Limitation
Import 65,000 Rows	9X	8X	ACE Driver Limitation
Export 1,000 Rows	13X	14X	ACE Driver Limitation
Export 10,000 Rows	37X	38X	ACE Driver Limitation
Export 65,000 Rows	33X	33X	ACE Driver Limitation

CONCLUSION

The introduction of new SAS/ACCESS products—including SAS/ACCESS Interface to Aster nCluster, SAS/ACCESS Interface to Greenplum, and SAS/ACCESS Interface to Sybase IQ—make available new data sources to use with SAS applications and solutions. Key features in these new data sources include bulk-data-movement capabilities, advanced features in existing engines, and specific DBMS options controlled by SAS. Careful study of SAS/ACCESS documentation could increase your ability to implement data extraction, data use, and data loading solutions.

Commitment to ongoing product enhancement has produced significant performance improvements in the PC Files Server in SAS/ACCESS Interface to PC File Formats and in SAS/ACCESS Interface to DB2 on z/OS. These performance improvements resulted in upgraded data movement in and usage optimization to current interfaces. The performance tests were performed in a shared R&D environment with decent hardware and a network. Depending on your environment, your performance tests might produce even better results than those in this paper.

REFERENCES

SAS Institute Inc., SAS/ACCESS documentation. Available at http://support.sas.com/documentation/onlinedoc/access/index.html.

SAS Institute Inc., Base SAS documentation. Available at http://support.sas.com/documentation/onlinedoc/base/index.html.

ACKNOWLEDGMENTS

I want to acknowledge several members of the SAS/ACCESS team who were key in the development of SAS/ACCESS software and contributed to the technical review of this paper:

Mauro Cazzari—SAS/ACCESS Interface to DB2 on z/OS and SAS/ACCESS Interface to Greenplum

Chris DeHart—SAS/ACCESS Interface to Sybase IQ

Keith Handlon—SAS/ACCESS Interface to Aster nCluster

Joe Schluter—PC Files Server in SAS/ACCESS Interface to PC File Formats

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Howard Plemmons
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Howard.Plemmons@sas.com